Reconstruction of offsets of an electron gun using deep learning and an optimization algorithm

David Meier[a], Gregor Hartmann[a], Jens Völker[a], Jens Viefhaus[a], and Bernhard Sick[b]

[a]Helmholtz-Zentrum Berlin für Materialien und Energie GmbH, Berlin, Germany
[b]Universität Kassel, Kassel, Germany

ABSTRACT

A photoelectron gun cavity is an electron source that every Energy Recovery Linac (ERL) requires. Its characteristics and precision determine the capabilities and performance of the ERL. Calibration of the electron gun is a crucial part of its initial setup, which requires a lot of time and experience. We present the first steps towards a tool that guides the electron gun operator, which knobs to turn in which position to achieve the desired properties. In this report, we determine the - typically difficult to identify - offsets between the simulation and the real-world device. We accomplish this by using machine learning and a global optimization algorithm.

Keywords: Deep Learning, Global Optimization, Machine Learning, Offsets, Electron Gun

1. INTRODUCTION

The Helmholtz-Zentrum Berlin (HZB) builds a new ERL demonstrator facility named bERLinPro (Berlin ERL Linac Prototype). It will provide a continuous-wave (CW) electron beam of high current (up to 100 mA as a single pass machine).[1]

A superconducting radiofrequency (SRF) photoelectron gun cavity is used as electron source.[2] This gun provides a direct pulsed electron beam and is crucial for the beam quality and the accelerator's performance. Even though the electron gun is only one part of the entire photoelectron injector of bERLinPro, its characteristics are crucial for the performance of the whole system.[3]

Another essential element is a superconducting solenoid magnet that focuses the extracted beam into the further accelerator elements and provides emittance compensation for the space charge dominated electron beam.[4] It is essential to align all elements of the photoinjector regarding the beam trajectory. Otherwise, the beam quality and thus, the performance of the whole accelerator are adversely affected.[5] The correct positioning is even more critical in the case of cryogenic systems because of the thermal expansion of the holders and the components themselves.[2]

The performance of the whole injector, including the alignment of all elements can be calculated numerically. One possible tool is the ASTRA (A Space Charge Tracking Algorithm) algorithm,[6] which is used in this survey. Single-particle tracking algorithms play an increasing role in the analysis of electron injectors.[3] They are used for designing and optimizing electron guns for different targets.[3, 7, 8] All relevant injector parameters (cavity and solenoid field values, offsets of the elements, and so on) are implemented in our simulation and result in a detailed phase-space distribution of a single electron bunch downstream of the injector.

The initial setup usually takes a lot of time, even for experienced engineers, due to the complexity of an electron gun. Sizeable effects on micrometer scales like small differences and errors can have huge impacts. Even worse, the inner processes of a photoinjector electron gun are non-deterministic and prone to positive feedback. The ultimate goal is to help the engineer finding the optimal settings of the electron gun required for the desired results. We want to define what results we should get, and we want to know which parameters we have to choose. In other words, we want to approximate the *inverse* of the simulation. One problem that occurs is that a simulation can never entirely reproduce the real world. Even small mistakes, e.g., wrong positions, can have a drastic impact on the results. As a first step, this study approximates the difference between the real world and the simulation.

For the numerical simulation of the photoelectron injector, a single particle tracking for different setups is performed, including the photocathode, the drive laser, the gun cavity with RF (radio frequency) and DC (direct current) fields and the solenoid magnet (six general parameters: laser pulse length, laser spot size, DC+RF field strength, emission phase, and solenoid focal strength). Additional settings in such a simulation are the individually geometrical offsets of those elements which are not well known in the real world (nine offset parameters: longitudinal cathode position, transverse laser offset, cavity field flatness, solenoid position, and angle offsets). The tracking algorithm calculates the electron paths through such an injector setup up to a virtual screen position downstream of the injector and outputs the phase-space distribution there. These distributions will be used to calculate the four arithmetic beam parameters (mean transverse positions, root mean square transverse beam size) and the average beam energy. All five values are also measurable in the real world at this position.

2. DATA CREATION

To find the offsets between the real device and our ideal simulation (without offsets), we use an optimization algorithm that frequently executes the simulation. However, one execution of an ASTRA simulation takes about five minutes on a current CPU. Thus, we require a fast approximation of the electron gun simulation. For this, an artificial neural network is used that will be described in section 3.

A sufficient amount of data is required to train a neural network. For this, the simulation is run 1 million times with a variation of the 15 input parameters shown in Table 1. Some of these parameter combinations result in non-trackable start-to-end simulations and are discarded. All these events are regarded as outliers and are left out of the training set. As a normalization method, feature scaling (min-max normalization) is used for input and simulation output.

Also, all simulation output values yielding output values outside acceptable parameter ranges (see Table 2) are not taken into account. The finally used dataset contains 546689 samples.

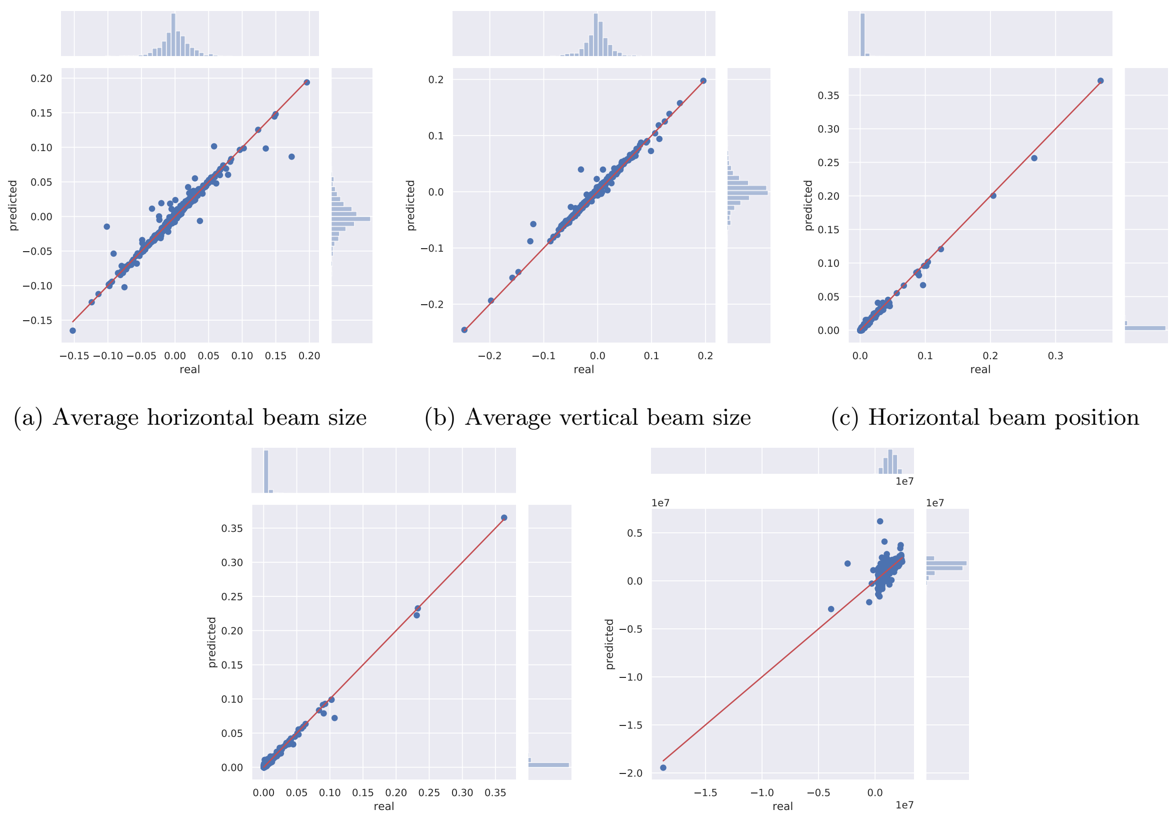
Table 1: Parameters and their ranges for the ASTRA simulations. All values are chosen from a uniform distribution of the specified interval. Fixed values: Bunch charge scale: 0.1 pC, solenoid position to cavity z-axis: 0.4625 m, stop position of tracking: 1.737 m, longitudinal offset of the input distribution like gun peak field but multiplied with 10^{-4}.

Parameter	Interval	Unit
Laser pulse length	$[0.6, 4] * 10^{-3}$	ns
Laser spot size on cathode	$[0.2, 0.8]$	mm
Gun peak field	$[9, 18]$	MV/m
Gun DC bias field	$[3, 5]$	kV
Cathode position	$[-20, -5]$	0.1 mm
Field flattnes	$[-0.5, 0.5]$	
Laser horizontal position	$[-1.5, 1.5]$	mm
Laser vertical position	$[-1.5, 1.5]$	mm
Solenoid horizontal position	$[-4, 4] * 10^{-3}$	mm
Solenoid vertical position	$[-4, 4] * 10^{-3}$	mm
Solenoid angle y-axis	$[-30, 30] * 10^{-3}$	rad
Solenoid angle x-axis	$[-30, 30] * 10^{-3}$	rad
Emission phase	$[-10, 70]$	deg
Solenoid strength	$[-0.1, 0.1]$	T

The resulting beam values at the virtual view screen variate over a wide range and can exceed the measurable range in the real world. For a better comparison between simulation and the real world, only those setups with beam parameters in the same range as the dimension of the actual view screen are used for further steps. These limits can be seen in Table 2.

Table 2: Calculated simulation or measured real-world output values.

Measured value	Acceptable parameter range	Unit
Average horizontal beam size	$[-30, 30]$	mm
Average vertical beam size	$[-30, 30]$	mm
Horizontal beam position	$[-30, 30]$	mm
Vertical beam position	$[-30, 30]$	mm
Average beam momentum	-	MeV/c



(a) Average horizontal beam size (b) Average vertical beam size (c) Horizontal beam position

(d) Vertical beam position (e) Average beam momentum

Figure 1: On the y-axis, the predicted values are shown; the x-axis shows the real simulation values. Ideally, all blue dots would lie on the red line. There is always shown a random subset of 1000 samples. If all samples were plotted, few outliers would conceal a lot of very accurately approximated samples.

3. APPROXIMATION OF THE SIMULATION

For the approximation of the simulation, we train a neural network. The training set is split into 60 % training data, 20 % test data, and 20 % validation data (for early stopping criterion). We use a simple five hidden layer network going up to 2002 neurons first, decreasing logarithmically to 447, 100, 22, and finally returning 5 outputs. A ReLU activation layer is used except for the output layer, and batch normalization is used in every layer (except for the output layer). As an optimization method, Adam (an adaptive learning rate optimization algorithm by Kingma et al.[9]) is used, the loss function is MSE (mean squared error). The hyperparameters that turned out to be best for training are a batch size of 2048 and a learning rate 0.001. In Figure 2 is shown a comparison of three different learning rates. This batch size is chosen for fast convergence. With a smaller batch size and learning rate, the approximation accuracy could likely be increased further. The MSE achieved on the test set with this network is about $1.13 * 10^{-5}$.

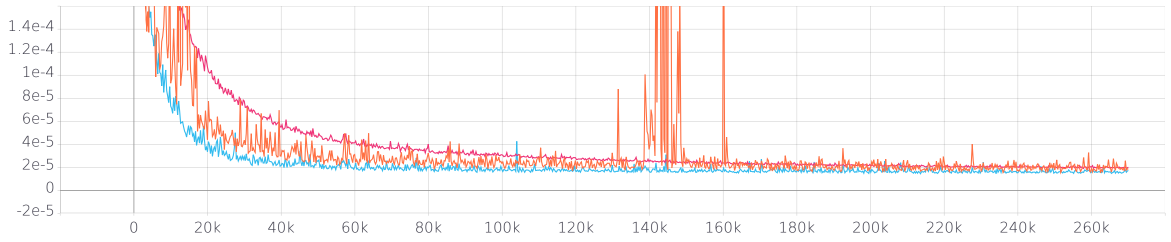


Figure 2: Comparison of the MSE (y-axis) of the validation set of three different learning rates: The light blue graph shows the MSE with learning rate 0.001, the orange graph with 0.01, and the magenta graph with 0.0001. The x-axis shows the time step (every batch is counted as one time step).

The results are depicted in Figure 1. The average beam sizes and beam positions are predicted very well, the average beam momentum acceptably well. The outliers might result from an insufficient particle amount during dataset creation. Thus, these points would be unpredictable.

4. APPROXIMATION OF OFFSETS

To approximate the offsets, we first add randomly generated offsets that we want to predict later. We define the inputs in our simulation without offsets as $x_{\text{simulation}} \in \mathbb{R}^{14}$, the inputs as they are in reality with $x_{\text{experiment}} \in \mathbb{R}^{14}$. The difference of these we call the offset $x_{\text{off}} \in \mathbb{R}^{14}$.

$$x_{\text{simulation}} = x_{\text{experiment}} + x_{\text{off}}$$

The approximation of the simulation is denoted by $y_{\text{approx}} : \mathbb{R}^{14} \to \mathbb{R}^5$. The real outputs in our experiments are called $y_{\text{experiment}} : \mathbb{R}^{14} \to \mathbb{R}^5$, they are shifted by an unknown offset. Again, we define the resulting difference as $y_{\text{off}} \in \mathbb{R}^5$.

$$y_{\text{approx}}(x) = y_{\text{experiment}}(x) + y_{\text{off}}$$

When having n measured values of our machine, we denote the i-th evaluation of our experiment with $y_{\text{experiment}}(x^{(i)}_{\text{experiment}})$ with $i \in \{1, \dots, n\}$. Furthermore, we know the setup of the machine $x^{(i)}_{\text{experiment}}$. To summarize, we already know $x^{(i)}_{\text{experiment}}$, $y_{\text{experiment}}(x^{(i)}_{\text{experiment}})$ and $y_{\text{approx}}(\cdot)$ in advance.

In order to determine the offsets x_{off} and y_{off}, we have to solve the following optimization problem:

$$\min_{x_{\text{off}}, y_{\text{off}}} \mathcal{L}(x_{\text{off}}, y_{\text{off}}) \tag{1}$$

with

$$\mathcal{L}(x_{\text{off}}, y_{\text{off}}) = \sum_{i=1}^{n} \left(y_{\text{approx}} \left(x^{(i)}_{\text{simulation}} \right) - y_{\text{experiment}} \left(x^{(i)}_{\text{experiment}} \right) \right)^2 \tag{2}$$

$$= \sum_{i=1}^{n} \left(y_{\text{experiment}} \left(x^{(i)}_{\text{experiment}} + x_{\text{off}} \right) + y_{\text{off}} - y_{\text{experiment}} \left(x^{(i)}_{\text{experiment}} \right) \right)^2 \tag{3}$$

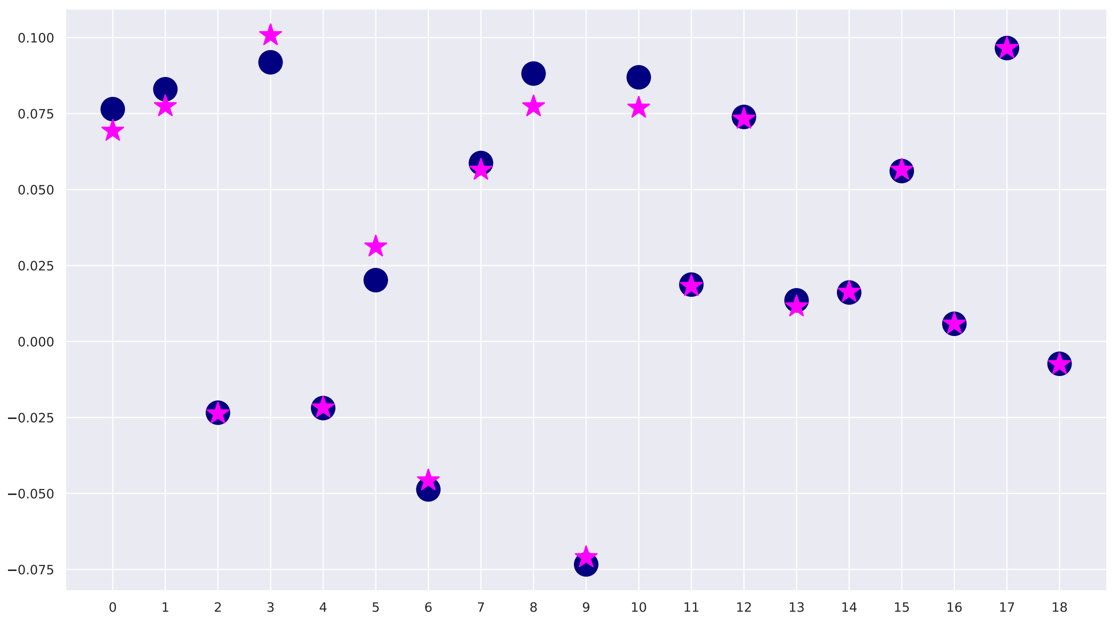


Figure 3: The result of the basin-hopping algorithm on finding the offsets with a maximum deviation of 0.1 after 20000 steps; the blue dots show the randomly generated offsets. The magenta stars show the approximation by the basin-hopping algorithm. The y-axis shows the deviation from the simulation value, the x-axis is the feature ID. Features 0 to 13 are the input values offsets x_{off}, 14 to 18 are the screen values offsets y_{off}. See Table 1 and Table 2 for the according feature and output names.

Basin-hopping is a stochastic global optimization algorithm that works by walking at first one step randomly. At this step, a local minimization algorithm is run. If the result is better than the previous step, it is accepted, and a new step is executed. If the result is worse, the step is getting discarded, and another step is chosen randomly. The acceptance criterion is more precisely the Metropolis criterion of the standard Monte Carlo algorithm.[10] The algorithm is presented with more details by Wales et al.[11]

The hyperparameters used in this study:

- Iterations: 20000
- Batch size: 128, the set of samples probed for acceptance criterion
- Starting point: $x_0 = \mathbf{0}$
- Temperature for acceptance criterion: 1.0

The results are shown in Figure 3 and Figure 4. The maximum deviation of offsets to find is set to 0.25 (a maximum deviation of ±25 %) and 0.1 (a maximum deviation of ±10 %). The used hyperparameters and results are compared in Table 3. Please note that basin-hopping is a stochastic algorithm, and finding the global minimum depends on the seed and starting point. For both parameters, the global minimum is not found yet (since the error is not 0), but it is approximated with an adequate deviation.

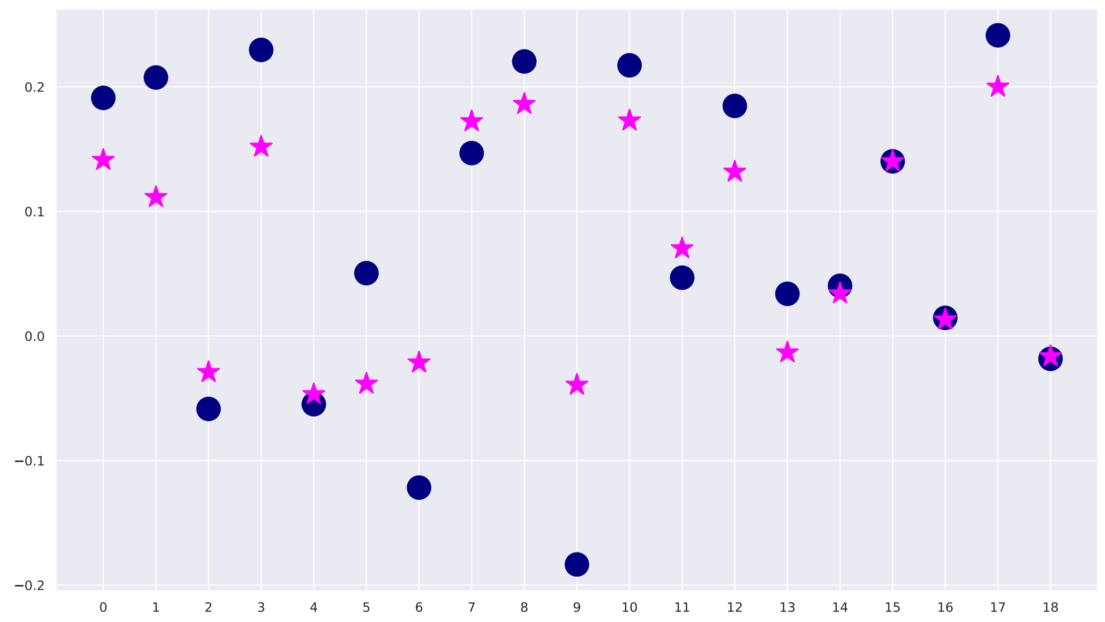


Figure 4: The result of the basin-hopping algorithm on finding the offsets with a maximum deviation of 0.25 after 20000 steps; the blue dots show the randomly generated offsets. The magenta stars show the approximation by the basin-hopping algorithm. The y-axis shows the deviation from the simulation value; the x-axis is the feature ID. Features 0 to 13 are the input values offsets x_{off}, 14 to 18 are the screen values offsets y_{off}. See Table 1 and Table 2 for the according feature and output names.

Table 3: Results and hyperparameters of experiments with different maximum deviations; Interval is the number of steps to wait until step size is updated. The squared error is calculated as described in Equation 2.

Maximum deviation	Stepsize	Inverval	Squared Error
0.1	0.001	100	$8.26 * 10^{-5}$
0.25	0.0025	500	0.2424

5. OUTLOOK

The approximation of the electron gun simulation achieves sufficient accuracy, as discussed in section 3. Furthermore, the offsets between simulation and reality can be determined reasonably with the used basin-hopping approach. Other approaches could be beneficial and should be investigated, e.g., neural networks, brute force algorithms, or a grid search. The next step towards application at the real device is solving the inversion of the simulation for optimum parameter estimation.

REFERENCES

[1] Neumann, A. et al., “The BERLinPro SRF Photoinjector System - From First RF Commissioning to First Beam,” in [*Proc. 9th International Particle Accelerator Conference (IPAC'18), Vancouver, BC, Canada, April 29-May 4, 2018*], *International Particle Accelerator Conference*, 1660–1663, JACoW Publishing, Geneva, Switzerland (June 2018). https://doi.org/10.18429/JACoW-IPAC2018-TUPML053.

[2] Kourkafas, G., Jankowiak, A., Kamps, T., Li, J., Schebek, M., and Völker, J., "Solenoid Alignment for the SRF Photoinjector of BERLinPro at HZB," in [*Proc. of International Particle Accelerator Conference (IPAC'17), Copenhagen, Denmark, 14-19 May, 2017*], *International Particle Accelerator Conference*, 1778–1780, JACoW, Geneva, Switzerland (May 2017). https://doi.org/10.18429/JACoW-IPAC2017-TUPIK042.

[3] Hosseinzadeh, M. and Sadighzadeh, A., "Design and numerical simulation of thermionic electron gun," *Chinese Physics C* **40** (September 2015).

[4] Serafini, L. and Rosenzweig, J. B., "Envelope analysis of intense relativistic quasilaminar beams in rf photoinjectors:ma theory of emittance compensation," *Phys. Rev. E* **55**, 7565–7590 (June 1997).

[5] Kuske, B. and Rudolph, J., "Beam Positioning Concept and Tolerance Considerations for BERLinPro," in [*Proc. 5th International Particle Accelerator Conference (IPAC'14)*], *International Particle Accelerator Conference*, 1105–1107, JACoW, Geneva, Switzerland (July 2014). https://doi.org/10.18429/JACoW-IPAC2014-TUPRO038.

[6] Flöttmann, K., [*Astra – A Space Charge Tracking Algorithm*], DESY, Hamburg, Germany, version 3.2 ed. (March 2017).

[7] Lewis, B. M., Tran, H. T., Read, M. E., and Ives, R. L., "Design of an electron gun using computer optimization," *IEEE Transactions on Plasma Science* **32**(3), 1242–1250 (2004).

[8] Pudjorahardjo, D. S., Suprapto, Darsono, Anwar, F., and Warseno, A. S., "Simulation study of electron beam spot size from thermionic electron gun using simion 8.1 software," *AIP Conference Proceedings* **2014**(1), 020156 (2018).

[9] Kingma, D. P. and Ba, J., "Adam: A method for stochastic optimization," in [*3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*], Bengio, Y. and LeCun, Y., eds. (2015).

[10] Li, Z. and Scheraga, H. A., "Monte carlo-minimization approach to the multiple-minima problem in protein folding," *Proceedings of the National Academy of Sciences* **84**(19), 6611–6615 (1987).

[11] Wales, D. J. and Doye, J. P. K., "Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms," *The Journal of Physical Chemistry A* **101**(28), 5111–5116 (1997).